

Práctica 3. Integración, entrega y despliegue continuo



Ampliación de Ingeniería del Software

Curso 2020-2021

Fátima Ezahra Smounat Mahidar

Milagros Mouriño Ursul

ÍNDICE

1. Url del repositorio de Github	3
2. Url de la aplicación desplegada en Heroku	3
3. Justificación de extensión de la memoria	3
4. Funcionamiento de los workflows	3
4.1. Feature-workflow	3
4.2. Push-develop-workflow	3
4.3. Nightly-workflow	4
4.4. Release-workflow	4
4.5. Master-workflow	5
5. Desarrollo de la nueva funcionalidad	6

1. Url del repositorio de Github

<https://github.com/mila564/ais-fe.smounat-m.mourino-urjc-2021>

2. Url de la aplicación desplegada en Heroku

<https://ais-fesmounatmmourino-2021.herokuapp.com/>

3. Justificación de extensión de la memoria

Hemos considerado necesario extender el número de páginas de la presente memoria para una mayor comprensión de la misma. La amplitud del trabajo se reduciría al exigido si no se hubiesen introducido imágenes, pero hemos preferido conservarlas, ya que se explica mejor el proceso que se ha llevado a cabo, sobre todo en el desarrollo de la nueva funcionalidad. Sin embargo, consideramos que el texto explica de manera concisa tanto el funcionamiento de los workflows como cada uno de los pasos para incorporar el servicio de LineBreaker.

4. Funcionamiento de los workflows

4.1. Feature-workflow

- Evento: se ejecuta ante el evento pull request (desde cualquier rama) a develop.
- Job: Build
- Steps:
 1. Clona el repositorio: Descarga el proyecto de Github dentro del runner.
 2. Instala Java 15: Consigue Java 15 en el runner.
 3. Ejecuta pruebas unitarias: Se verifica si los tests unitarios pasan dentro del runner.
 4. Ejecuta pruebas de API REST: Se comprueba que los tests de Rest pasan en el runner.

```
! feature-workflow.yml X
github > workflows > ! feature-workflow.yml
1 |name: feature-workflow
2
3 |on:
4 |  pull_request:
5 |    branches:
6 |      - develop
7
8 |jobs:
9 |  build:
10 |    name: Build Java application
11 |    runs-on: ubuntu-20.04
12 |    steps:
13 |      - name: Clone the repository
14 |        uses: actions/checkout@v2 1
15
16 |      - name: Set up java version
17 |        uses: actions/setup-java@v1 2
18 |        with:
19 |          java-version: 15
20
21 |      - name: Run Unitary tests
22 |        run: mvn -B '-Dtest=es.urjc.code.daw.library.unitary.*Test' test 3
23
24 |      - name: Run Rest tests
25 |        run: mvn -B '-Dtest=es.urjc.code.daw.library.e2e.rest.*Test' test 4
```

4.2. Push-develop-workflow

- Evento: se produce ante el evento push en la rama de integración.
- Job: Build
- Steps:
 - Los pasos del 1 al 4 son idénticos a los del workflow anterior.
- 5. Ejecuta pruebas de Selenium: Se llevan a cabo los tests de Selenium.

```
! push-develop-workflow.yml X
github > workflows > ! push-develop-workflow.yml
1 |name: push-develop-workflow
2
3 |on:
4 |  push:
5 |    branches:
6 |      - develop
7
8 |jobs:
9 |  build:
10 |    name: Build Java application
11 |    runs-on: ubuntu-20.04
12 |    steps:
13 |      - name: Clone the repository
14 |        uses: actions/checkout@v2 1
15
16 |      - name: Set up java version
17 |        uses: actions/setup-java@v1 2
18 |        with:
19 |          java-version: 15
20
21 |      - name: Run Unitary tests
22 |        run: mvn -B '-Dtest=es.urjc.code.daw.library.unitary.*Test' test 3
23
24 |      - name: Run Rest tests
25 |        run: mvn -B '-Dtest=es.urjc.code.daw.library.e2e.rest.*Test' test 4
26
27 |      - name: Run Selenium tests
28 |        run: mvn -B '-Dtest=es.urjc.code.daw.library.e2e.selenium.*Test' test 5
29
```

- Nota: Hemos tenido que configurar el navegador para que arranque en modo headless.

```
@BeforeEach
public void setupTest() {
    String host = System.getProperty("host", "localhost");
    ChromeOptions chromeOptions = new ChromeOptions();
    chromeOptions.addArguments("--headless");
    this.driver = new ChromeDriver(chromeOptions);
    this.wait = new WebDriverWait(driver, 10);
}
```

4.3. Nightly-workflow

- Evento: se ejecuta todas las noches aproximadamente a las 00:00 A.M (UTC).
- Job: Build
- Steps:
 - Los pasos del 1 al 5 son idénticos a los del workflow anterior.
- 6. Genera el artefacto: Compila y origina el JAR (sin volver a ejecutarse los tests).
- 7. Sube el JAR: Se carga el artefacto en GitHub para poder generar la imagen Docker en el siguiente job.
- Job: Publish in DockerHub
- Steps:
 - El paso 1 es análogo a los de los jobs anteriores (se clona el repositorio).
- 2. Descarga el artefacto: Baja el JAR que cargamos en el job anterior bajo el nombre de target.
- 3. Genera una etiqueta: Se crea una etiqueta denominada tag que contendrá la fecha del día utilizada para construir la imagen Docker.
- 4. Inicia sesión en Docker Hub: Login en Docker Hub con los secretos registrados en las configuraciones del repositorio.
- 5. Construye la imagen: Crea la imagen Docker con el secreto que registra el nombre de usuario de Docker Hub y el tag generado en el paso 3.
- 6. Empujar la imagen al registro de Docker Hub: Se sube la imagen al registry de Docker Hub.

```
! nightly-workflow.yml
github > workflows > ! nightly-workflow.yml
1 name: nightly-workflow
2
3 on:
4   schedule:
5     - cron: '0 0 * * *'
6
7 jobs:
8   build:
9     name: Build Java application
10    runs-on: ubuntu-20.04
11    steps:
12      - name: Clone the repository
13        uses: actions/checkout@v2
14        with:
15          ref: develop
16
17      - name: Set up java version
18        uses: actions/setup-java@v1
19        with:
20          java-version: 15
21
22      - name: Run Unitary tests
23        run: mvn -B -Dtest=es.urjc.code.daw.library.unitary.*Test test
24
25      - name: Run Rest tests
26        run: mvn -B -Dtest=es.urjc.code.daw.library.e2e.rest.*Test test
27
28      - name: Run Selenium tests
29        run: mvn -B -Dtest=es.urjc.code.daw.library.e2e.selenium.*Test test
30
31      - name: Generate artifact
32        run: mvn -B package -DskipTests
33
34      - name: Upload jar for next job
35        uses: actions/upload-artifact@v2
36        with:
37          name: target
38          path: target
39          retention-days: 1
```

```
publish_in_dockerhub:
name: Publish in DockerHub
runs-on: ubuntu-20.04
needs: [build]
steps:
  - name: Clone the repository
    uses: actions/checkout@v2
  - name: Download Jar from previous job
    uses: actions/download-artifact@v1
    with:
      name: target
  - name: Generate tag
    run: echo ::set-output name=tag::$(date +%Y%m%d.%H%M%S)
    id: project
  - name: Login to DockerHub
    run: docker login -u ${{ secrets.DOCKERHUB_USERNAME }} -p ${{ secrets.DOCKERHUB_TOKEN }}
  - name: Build image
    run: docker build -t ${{ secrets.DOCKERHUB_USERNAME }}/books:dev-$( steps.project.outputs.tag ) .
  - name: Push image to DockerHub
    run: docker push ${{ secrets.DOCKERHUB_USERNAME }}/books:dev-$( steps.project.outputs.tag )
```

4.4. Release-workflow

- Evento: se desencadena con cada evento push sobre cualquier rama de release.
- Job: Build
- Steps:
 - Los pasos del job de build son iguales a los de push-develop-workflow (Apartado 3.2).

```
! release-workflow.yml
github > workflows > ! release-workflow.yml
1 name: release-workflow
2
3 on:
4   push:
5     branches:
6       - 'release/**'
7
8 jobs:
9   build:
10    name: Build Java application
11    runs-on: ubuntu-20.04
12    steps:
13      - name: Clone the repository
14        uses: actions/checkout@v2
15
16      - name: Set up java version
17        uses: actions/setup-java@v1
18        with:
19          java-version: 15
20
21      - name: Run Unitary tests
22        run: mvn -B -Dtest=es.urjc.code.daw.library.unitary.*Test test
23
24      - name: Run Rest tests
25        run: mvn -B -Dtest=es.urjc.code.daw.library.e2e.rest.*Test test
26
27
```

4.5. Master-workflow

- Evento: se lleva a cabo cuando se produce un push de la rama master (al aceptarse el pull request proveniente de una rama de release).
- Job: Build
 - Steps:
 - El job de build y publish_in_dockerhub del master-workflow son idénticos a los de nightly-workflow (Apartado 3.3). La única diferencia es el tag con el que se genera la imagen Docker, el cual es la versión del pom.xml.
- Job: Publish in heroku registry
 - Steps:
 1. Clona el repositorio: Copia el contenido del repositorio en el runner.
 2. Baja el JAR: Descarga el artefacto de Github en el runner.
 3. Construye la imagen Docker: Se crea la imagen a partir de los comandos del Dockerfile y el JAR obtenido en el paso previo.
 4. Instalar el cliente de Heroku: Se obtiene el cliente de Heroku.
 5. Cambiar la imagen para que apunte al registro de Heroku: Se sustituye el tag books por el formato exigido por Heroku.
 6. Login en el registry de Heroku: Inicia sesión en el registry de Heroku.
 7. Empuja imagen al registry de Heroku: Sube la imagen al contenedor de imágenes de Heroku.
- Job: Deploy to Heroku:
 - Steps:
 1. Clona el repositorio: Se coloca el contenido del proyecto en el runner.
 2. Instala el cliente de Heroku: Se obtiene el cliente de Heroku.
 3. Login en Heroku: Se loguea en el registry de Heroku.
 4. Despliega en Heroku: Se despliega la aplicación en los servidores de Heroku.
 5. Ejecuta los test de Rest: Se verifica que los tests de API REST siguen pasando en el servidor donde se despliega la aplicación.

```
66 publish_in_heroku_registry:
67   name: Publish in Heroku Registry
68   runs-on: ubuntu-20.04
69   needs: [build]
70   env:
71     HEROKU_API_KEY: ${ secrets.HEROKU_API_KEY }
72     HEROKU_APP: ${ secrets.HEROKU_APP }
73   steps:
74     - name: Clone repository 1
75       uses: actions/checkout@v2
76
77     - name: Download Jar from previous job 2
78       uses: actions/download-artifact@v1
79       with:
80         name: target
81
82     - name: Build Docker Image 3
83       run: docker build -t books .
84
85     - name: Install Heroku CLI 4
86       run: curl https://cli-assets.heroku.com/install-ubuntu.sh | sh
87
88     - name: Change image name to point to Heroku registry 5
89       run: docker tag books registry.heroku.com/$HEROKU_APP/web
90
91     - name: Login in Heroku container registry 6
92       run: heroku container:login
93
94     - name: Push image to Heroku Registry 7
95       run: docker push registry.heroku.com/$HEROKU_APP/web
96
```

```
97 deploy_to_heroku:
98   name: Deploy to Heroku
99   runs-on: ubuntu-20.04
100   needs: [publish_in_heroku_registry]
101   env:
102     HEROKU_API_KEY: ${ secrets.HEROKU_API_KEY }
103     HEROKU_APP: ${ secrets.HEROKU_APP }
104   steps:
105     - name: Clone repository 1
106       uses: actions/checkout@v2
107
108     - name: Install Heroku CLI 2
109       run: curl https://cli-assets.heroku.com/install-ubuntu.sh | sh
110
111     - name: Login in Heroku 3
112       run: heroku container:login
113
114     - name: Deploy to Heroku 4
115       run: heroku container:release -a $HEROKU_APP web
116
117     - name: Run Rest tests 5
118       run: mvn -B -Dhost=$host -Dtest=es.urjc.code.daw.library.e2e.rest.*Test test
119
120     - name: Run Selenium tests 6
121       run: mvn -B -Dhost=$host -Dtest=es.urjc.code.daw.library.e2e.selenium.*Test test
122
```

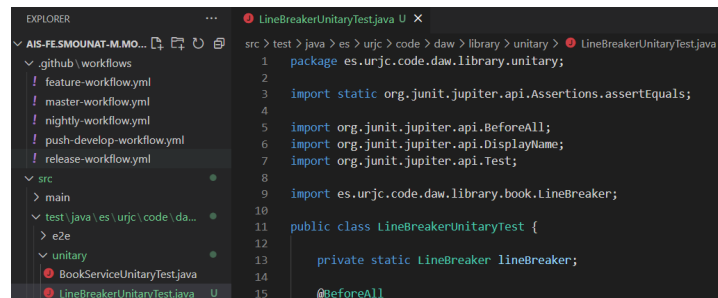
6. Ejecuta los test de Selenium: Se comprueba que las pruebas de Selenium no fallan en la aplicación desplegada.

5. Desarrollo de la nueva funcionalidad

Creamos la rama de **feature** para desarrollar la funcionalidad exigida:

```
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git branch
* develop
master
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git checkout -b feature/feature-1 develop
Switched to a new branch 'feature/feature-1'
```

Añadimos en **src\test\java\es\urjc\code\daw\library\unitary**, la clase **LineBreakerUnitaryTest.java**:



The screenshot shows the VS Code Explorer on the left with the file structure: `src > test > java > es > urjc > code > daw > library > unitary`. The file `LineBreakerUnitaryTest.java` is highlighted. The Editor on the right shows the code for `LineBreakerUnitaryTest.java`:

```
1 package es.urjc.code.daw.library.unitary;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4
5 import org.junit.jupiter.api.BeforeAll;
6 import org.junit.jupiter.api.DisplayName;
7 import org.junit.jupiter.api.Test;
8
9 import es.urjc.code.daw.library.book.LineBreaker;
10
11 public class LineBreakerUnitaryTest {
12
13     private static LineBreaker lineBreaker;
14
15     @BeforeAll
```

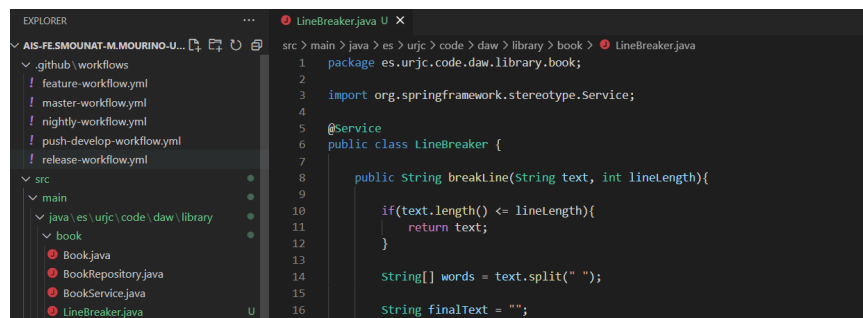
Con **git status** observamos el estado del directorio de trabajo. El fichero **LineBreakerUnitaryTest.java** está en estado **untracked**. Lo añadimos al área de staging con **git add**. Al ejecutarse en la terminal el comando **git status**, observamos que se ha añadido adecuadamente (al pintarse de verde). Con **git commit**, los nuevos cambios del proyecto se actualizan en el repositorio local. Mediante el parámetro **-m**, se especifica el cometido del commit ('Add LineBreakerUnitaryTest.java').

```
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git status
On branch feature-1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/test/java/es/urjc/code/daw/library/unitary/LineBreakerUnitaryTest.java

nothing added to commit but untracked files present (use "git add" to track)
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git add src/test/java/es/urjc/code/daw/library/unitary/LineBreakerUnitaryTest.java
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git status
On branch feature-1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   src/test/java/es/urjc/code/daw/library/unitary/LineBreakerUnitaryTest.java

PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git commit -m "Add LineBreakerUnitaryTest.java"
[feature/feature-1 9f31437] Add LineBreakerUnitaryTest.java
1 file changed, 123 insertions(+)
create mode 100644 src/test/java/es/urjc/code/daw/library/unitary/LineBreakerUnitaryTest.java
```

Repetimos el mismo procedimiento para añadir **LineBreaker.java** que está en **src\main\java\es\urjc\code\daw\library\book** en un nuevo commit con los comandos explicados anteriormente:



The screenshot shows the VS Code Explorer on the left with the file structure: `src > main > java > es > urjc > code > daw > library > book`. The file `LineBreaker.java` is highlighted. The Editor on the right shows the code for `LineBreaker.java`:

```
1 package es.urjc.code.daw.library.book;
2
3 import org.springframework.stereotype.Service;
4
5 @Service
6 public class LineBreaker {
7
8     public String breakLine(String text, int lineLength){
9
10         if(text.length() <= lineLength){
11             return text;
12         }
13
14         String[] words = text.split(" ");
15
16         String finalText = "";
```

```

On branch feature/feature-1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/main/java/es/urjc/code/daw/library/book/LineBreaker.java

nothing added to commit but untracked files present (use "git add" to track)
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git add src/main/java/es/urjc/code/daw/library/book/LineBreaker.java
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git status
On branch feature/feature-1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/main/java/es/urjc/code/daw/library/book/LineBreaker.java

PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git commit -m "Add LineBreaker.java"
[feature/feature-1 8524f54] Add LineBreaker.java
1 file changed, 63 insertions(+)
create mode 100644 src/main/java/es/urjc/code/daw/library/book/LineBreaker.java

```

Una vez creados los dos commits, ejecutamos el comando **git push origin feature/feature-1** para empujar los cambios al repositorio remoto:

```

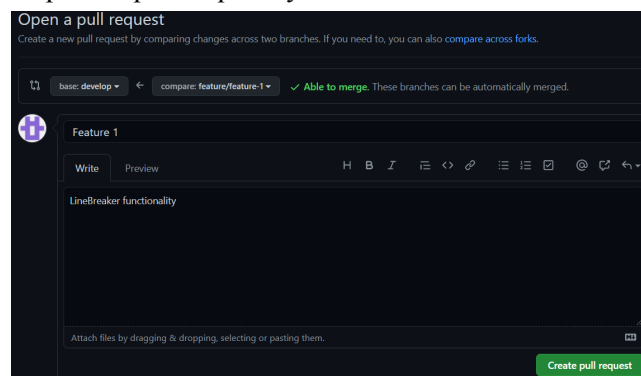
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git push origin feature/feature-1
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (24/24), 2.50 KiB | 320.00 KiB/s, done.
Total 24 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
remote:
remote: Create a pull request for 'feature/feature-1' on GitHub by visiting:
remote:   https://github.com/mila564/ais-fe.smounat-m.mourino-urjc-2021/pull/new/feature/feature-1
remote:
To https://github.com/mila564/ais-fe.smounat-m.mourino-urjc-2021.git
 * [new branch]      feature/feature-1 -> feature/feature-1

```

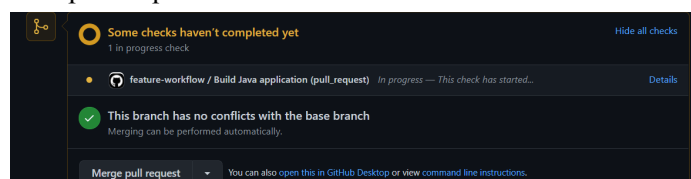
Ahora, tenemos que desencadenar el primer workflow. Esto se produce antes de integrar los cambios de la nueva funcionalidad dentro de la rama de integración (develop) mediante un pull request:



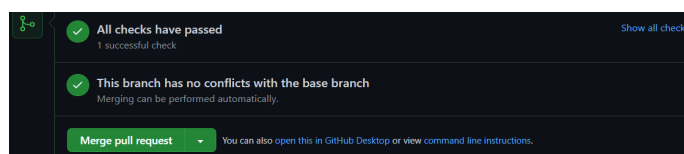
Hacemos clic en ‘Compare & pull request’ y rellenamos la información:



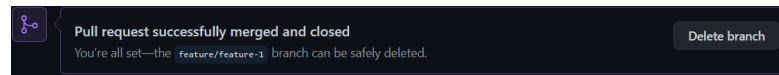
Pinchamos en ‘Create pull request’:



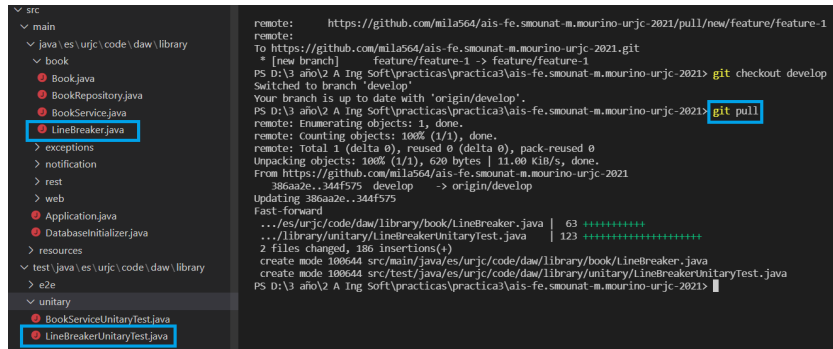
Antes de integrar el código en la rama develop, se construye la build (del feature-workflow) y se pasan los tests (Unitarios y Rest) mediante el servidor de integración continua (CI) que proporciona Github:



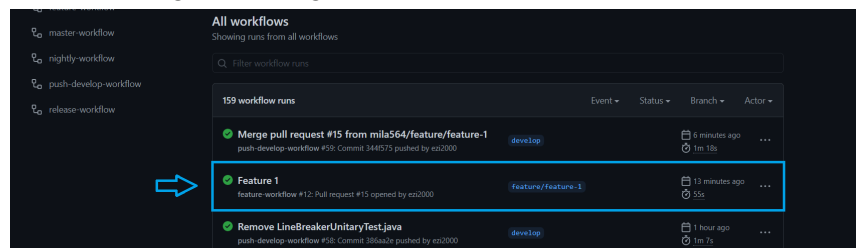
El job de build se ejecuta correctamente por lo que damos clic en ‘Merge pull request’ y ‘Confirm merge’ para integrar los cambios en la rama de integración.



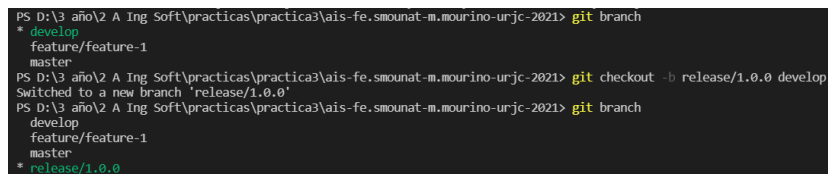
Hacemos un **git pull** en nuestra rama develop para bajar los cambios del servidor remoto:



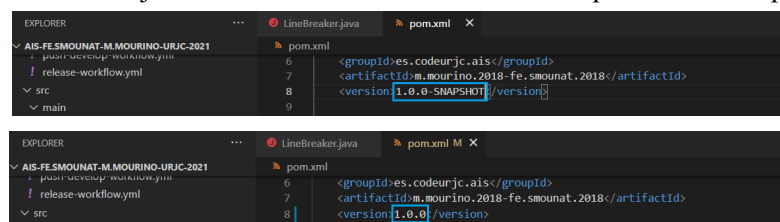
Cabe destacar, que empezamos a aplicar git-flow a partir del workflow de 'Feature 1', tal y como se muestra en la siguiente imagen:



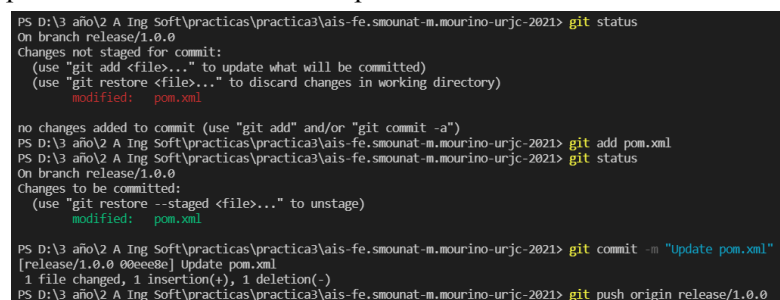
El push-develop-workflow se desencadena, después del de Feature 1, al haber aceptado el pull request anterior, ya que el commit de merge se traduce en un push. Creamos la rama de release en la que prepararemos la versión que sacaremos a producción:



Debemos suprimir el sufijo '-SNAPSHOT' de la versión de la aplicación en el pom.xml:



Subimos al repositorio remoto los cambios producidos en la rama de release:



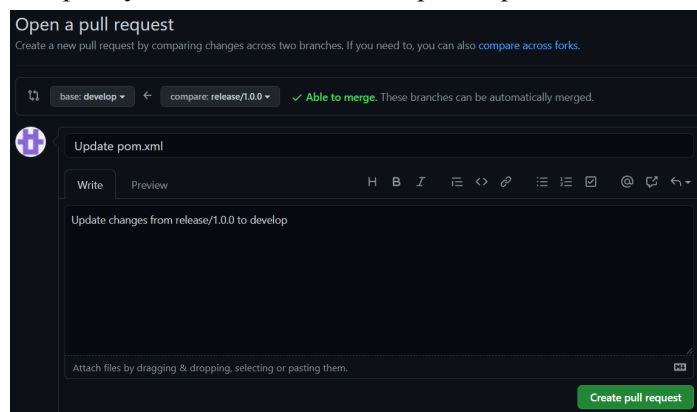
Como se ha hecho un push en la rama de release, se ejecuta el cuarto de los workflows definidos en la fase previa de la práctica, que verifica que los tests siguen pasando (Unitarios y Rest):



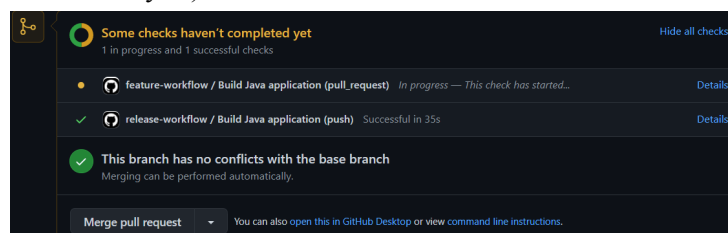
Tenemos que aumentar el minor version de la versión de la aplicación del pom.xml en develop. Para ello, lo primero que debemos hacer es mezclar mediante un pull request los cambios de la rama release/1.0.0 a develop. Damos clic en ‘Compare & pull request’:



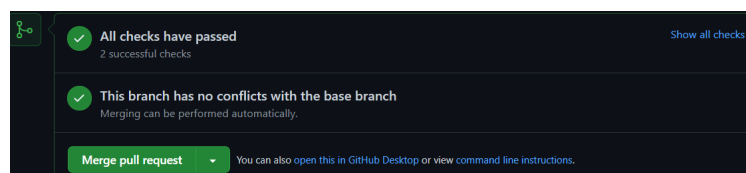
Preparamos el pull request y se hace clic en ‘Create pull request’:



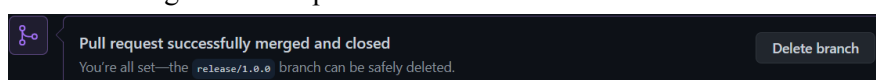
Como se realiza un **pull request** a develop, se ejecutará nuevamente el primer workflow definido (feature-workflow.yml):



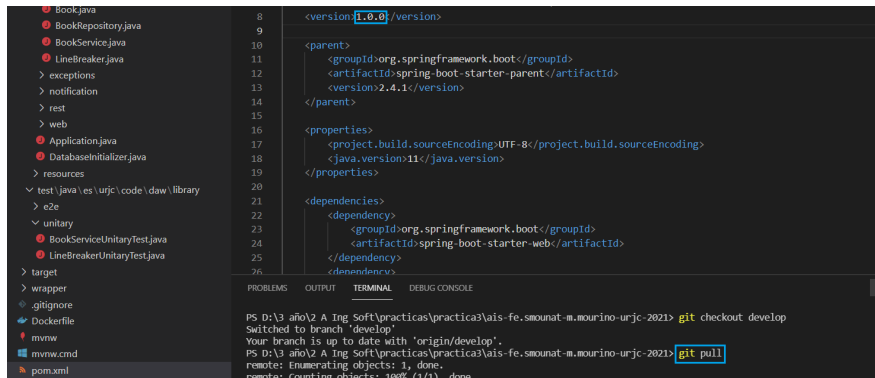
Observamos que no se rompe la build y damos clic en ‘Merge pull request’ y ‘Confirm merge’:



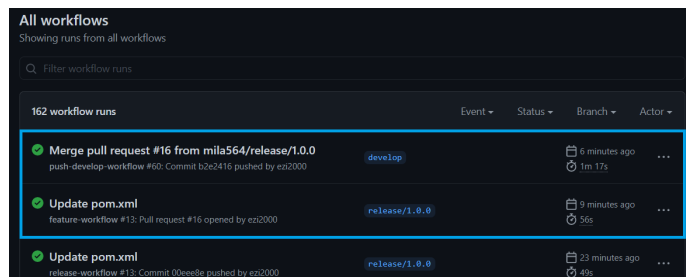
Se ha producido el merge en develop correctamente:



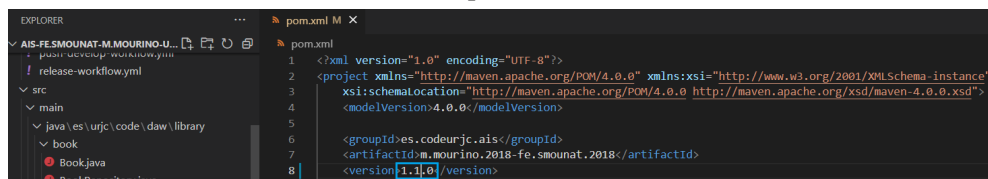
Bajamos los cambios de origen:



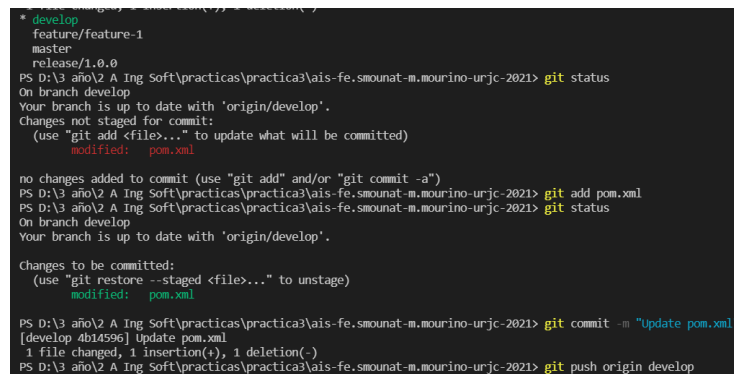
Observamos en la siguiente captura de pantalla que, por una parte, se ejecutó el feature-workflow al haber hecho el pull request desde la rama de release a develop y, por otra, que con el merge se ejecuta el push-develop-workflow:



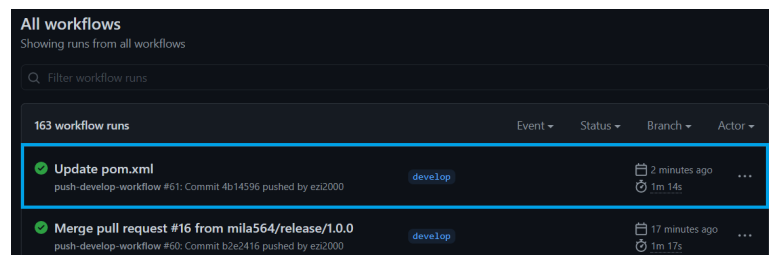
Ahora, aumentamos el menor version de la aplicación:



Subimos los cambios al repositorio remoto:



Comprobamos que, al hacerse el push en develop, se ejecuta de nuevo el push-develop-workflow:



Volvemos a colocarnos en la rama de release:

```
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git branch
* develop
  feature/feature-1
  master
  release/1.0.0
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git checkout release/1.0.0
Switched to branch 'release/1.0.0'
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smounat-m.mourino-urjc-2021> git branch
  develop
  feature/feature-1
  master
* release/1.0.0
```

Creamos el pull request:

Como los workflows anteriores se ejecutaron adecuadamente, se procede a aceptar el pull request a master:

Se realiza correctamente la operación:

Observamos que, al realizarse un commit de merge, se ejecuta adecuadamente el quinto workflow (master-workflow):

All workflows
Showing runs from all workflows

Filter workflow runs

164 workflow runs

	Event	Status	Branch	Actor
<div>✓ Merge pull request #17 from mila564/release/1.0.0</div> <div>master-workflow #43: Commit 686f998 pushed by ezi2000</div>	master	4 minutes ago	4m 12s	...
<div>✓ Update pom.xml</div> <div>push-develop-workflow #61: Commit 4b14596 pushed by ezi2000</div>	develop	3 hours ago	1m 14s	...

En Docker Hub se publica correctamente la imagen de la aplicación:

mila534/books

This repository does not have a description

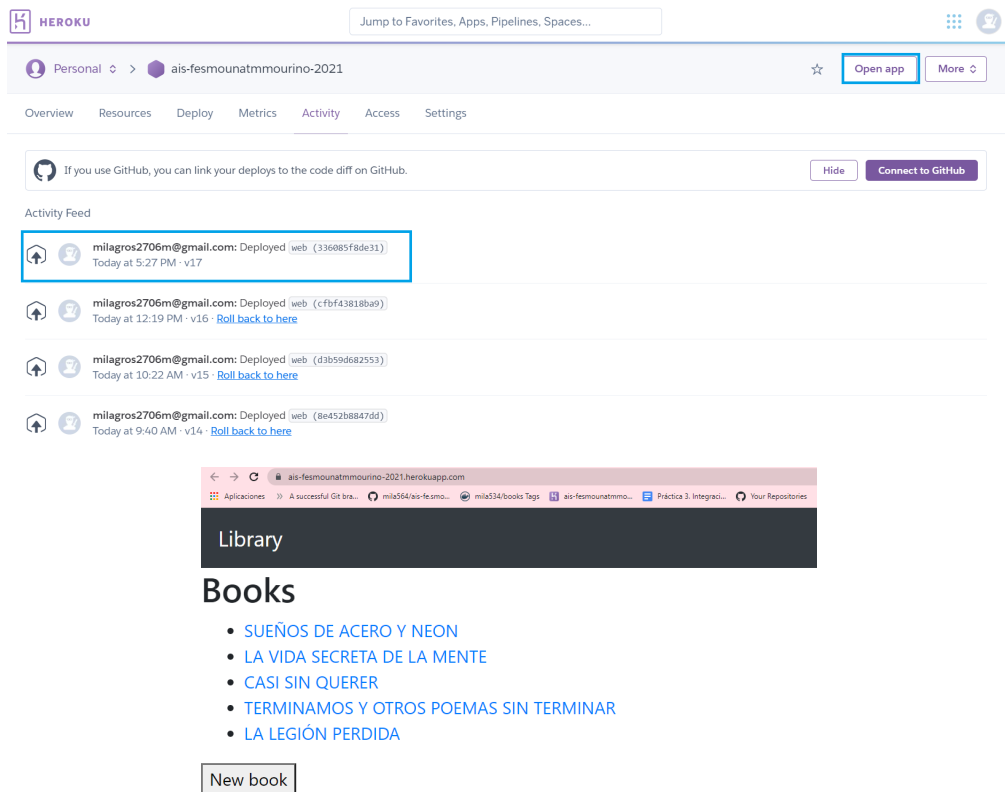
Last pushed: 23 minutes ago

Tags and Scans VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 12 tag(s).

TAG	OS	PULLED	PUSHED
1.0.0		23 minutes ago	23 minutes ago

Por otro lado, la aplicación se despliega en Heroku:



HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal > ais-fesmounatmmourino-2021

Overview Resources Deploy Metrics Activity Access Settings

If you use GitHub, you can link your deploys to the code diff on GitHub. [Hide](#) [Connect to GitHub](#)

Activity Feed

- milagros2706m@gmail.com: Deployed web (336085f8de311) Today at 5:27 PM · v17
- milagros2706m@gmail.com: Deployed web (cfbf43818b99) Today at 12:19 PM · v16 · [Roll back to here](#)
- milagros2706m@gmail.com: Deployed web (d3b59d682553) Today at 10:22 AM · v15 · [Roll back to here](#)
- milagros2706m@gmail.com: Deployed web (8e452b8847dd) Today at 9:40 AM · v14 · [Roll back to here](#)

ais-fesmounatmmourino-2021.herokuapp.com

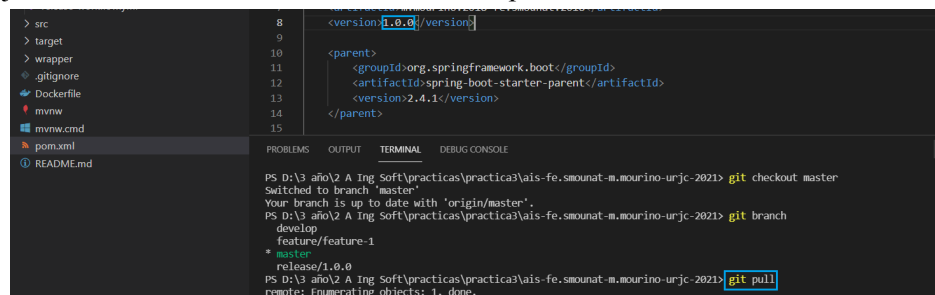
Library

Books

- SUEÑOS DE ACERO Y NEON
- LA VIDA SECRETA DE LA MENTE
- CASI SIN QUERER
- TERMINAMOS Y OTROS POEMAS SIN TERMINAR
- LA LEGIÓN PERDIDA

[New book](#)

Nos bajamos los cambios de la rama master del repositorio remoto al local:



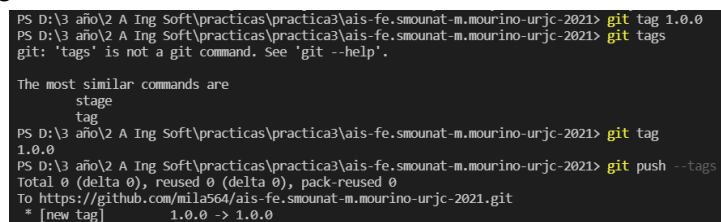
```
> src
> target
> wrapper
> .gitignore
> Dockerfile
> mvnw
> mvnw.cmd
> pom.xml
> README.md

8 <version>1.0.0</version>
9
10 <parent>
11   <groupId>org.springframework.boot</groupId>
12   <artifactId>spring-boot-starter-parent</artifactId>
13   <version>2.4.1</version>
14 </parent>
15

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git checkout master
Switched to branch 'master'
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git branch
develop
feature/feature-1
* master
release/1.0.0
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git pull
remote: Enumerating objects: 1, done.
```

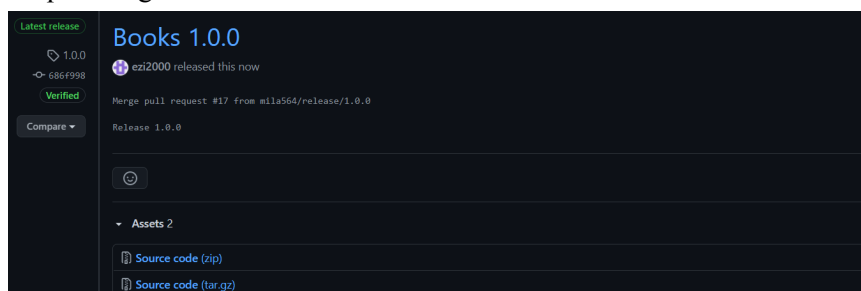
Asociamos un tag al commit de merge con el comando **git tag** y con **git push --tags** subimos la etiqueta a origin:



```
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git tag 1.0.0
git: 'tags' is not a git command. See 'git --help'.

The most similar commands are
stage
tag
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git tag
1.0.0
PS D:\3 año\2 A Ing Soft\practicas\practica3\ais-fe.smonat-m.mourino-urjc-2021> git push --tags
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mila564/ais-fe.smonat-m.mourino-urjc-2021.git
* [new tag]          1.0.0 -> 1.0.0
```

Observamos que el tag se ha asociado a la release:



Latest release

1.0.0

686f998

Verified

Compare

Books 1.0.0

ezl2000 released this now

Merge pull request #17 from mila564/release/1.0.0

Release 1.0.0

Assets 2

- Source code (zip)
- Source code (tar.gz)